Subject | " المحاضرة السادسة " Date |

البرامج الفرعية (الدوال):

أثناء كتابة برامج بلغة الـ C++ يمكن أن تصادفنا مشكلة تتمثل في وجود عدة أوامر أو تعليمات يتطلب تكرار تنفيذها أكثر من مرة.

- ولحل هذه المشكلة نقوم بإنشاء برنامج فرعي في مقدمة البرنامج، وذلك نقوم بإعطائه اسماً اختيارياً ويضم هذه البرنامج الفرعي جميع الأوامر المتكررة.

- يستخدم البرنامج الفرعي من أجل:

1 - اختصار البرنامج. 2 توفير الذاكرة.

3 - سرعة في التنفيذ 4 - سهولة كشف الأخطاء.

- هناك نوعان من البرامج الفرعية:

1 - برنامج فرعي يعيد قيمة (الدالة بلغة الباسكال).

2 - برنامج فرعي لا يعيد قيمة (الإجراء بلغة الباسكال).

- عند تنفيذ البرنامج ينتقل التنفيذ من البرنامج الرئيسي الى الدالة البرنامج الفرعي لينفذ الأوامر المتكررة.

- من أجل التعامل مع الدوال التي لا تعيد قيمة نستخدم الكلمة المحجوزة (Void).

- Void: هي نوع من أنواع المتغيرات والذي يعبر عن اللاشيء.

- لغة الـ C تتعامل مع الدوال فقط.

- الدوال التي تعيد قيمة نستخدم الأمر return (القيمة العائدة).

الصيغة: return (القيمة العائدة);

علماً أن يكون هذا الأمر هو آخر أمر في الدالة

- من أجل إنشاء الدالة نقوم بما يلي:

1 - [نحدد] نوع المتغيرات للدالة، وعند عدم تحديد النوع تأخذ الدالة من النوع (int)

2 - نضع اسماً اختيارياً للدالة يلي نوع المتغيرات.

3 - يلي اسم الدالة قائمة من الوسطاء (البارامترات) محصورة بين قوسين ( ).

- عند عدم وجود وسطاء تترك الأقواس فارغة.

4 - نكتب أوامر الدالة محصورة بين القوسين { أوامر الدالة } علماً أن يكون آخر أمر في الدالة هو / return إذا كان المطلوب من الدالة إعادة قيمة.

- ملخص: نوع المتغيرات اسم الدالة (نذكر الوسطاء إن وجدت)

Subject Date

(نذكر المعطيات ان وجدت) اسم الدالة ← نوع المتغيرات

```
{
   ________
   ________
   ________
   return( القيمة ) ;
}
```

مثال

```
int   MM(int x , int y)
{
   int a ;      ← متغير
   float b ;
   ;
   return( القيمة ) ;
}
void   ss( char c)
{
   int k ;
   ;
}
```

- عند كتابة البرنامج يتضمن عدة برامج فرعية نقوم في مقدمة البرنامج بتعريف البرامج الفرعية وفي الغالب يتم شرح البرامج الفرعية في نهاية البرنامج الرئيسي.

مثال

```
#include <iostream.h>
int   MM( int x , int y);     } برامج فرعية
void   ss(float a , char c) ;  }
void   main ( )
{
```

Subject .... Date ....

```
int k1, k2;
float f1, f2;
    ...
    القيمة = MM(   );
        SS(   );
}
int MM(int x, int y) X
{
    ...
    return(القيمة);
}
void SS(float a, char c) X
{
    ...
}
```

لا نضع فاصلة منقوطة لأنه خرج

خرج من برنامج الفرعي الأول

خرج

خرج من برنامج الفرعي الثاني

خرج من لا نضع فاصلة منقوطة

* البارامترات الفعلية والبارامترات الشكلية:

```
int MM(int x, int y);
```

وسائط شكلية

```
t = MM(5, 10);
```

← إسناد لمتغير

وسائط فعلية

```
void SS(float a, char c);
```

براميترات شكلية

```
SS(1.5, 'A');
```

← لا نسند قيمة لأنها من نوع void

براميترات فعلية

عند كتابة البرنامج الفرعي نضع له اسماً اختيارياً يليه قائمة من الوسطاء أو البارامترات محصورة بين قوسين ( ) ، هذه البارامترات تسمى بارامترات شكلية ، أما عند استدعاء الدالة من داخل البرنامج الرئيسي يتم ذلك بذكر اسم الدالة تليه قائمة من الوسطاء

Subject | Date |

محاضرة ... ملحوظات

محصورة بين قوسين ( ) ، وهذه البارامترات تسمى بارامترات فعلية

- يجب أن يحصل تطابق وتوافق بين البارامترات الفعلية والشكلية من حيث العدد والنوع

- ملاحظات:

[1] - لا يجوز تعريف دالة داخل دالة أخرى.

[2] - يجب أن يحصل توافق بين نوع الدالة ونوع القيمة العائدة بالأمر return.

- عند التعامل مع الدوال الحرفية من نوع char يكون الاستدعاء أيضاً مع رقم الترتيب للمعرف بالدالة.

- تمرين: باستخدام مفهوم البرامج الفرعية، اكتب برنامج يقوم بما يلي:
(١) جمع عددين صحيحين (٢) القيمة الصغرى للعددين الصحيحين.
(٣) طباعة العبارة Borland c++ ... مرات.

```
# include <iostream.h>
int sum(int a, int t);
int min(int c1, int c2);
void show( );
void main( )
{
  int x, y, s, m;
  cin>>x >> y;
  s = sum(x,y); cout<<"\n s="<<s;
  m = min(x,y); cout<<"\n m="<<m;
  show( );
}
```

البرامج الفرعية:

```
int sum(int a, int t)
{ int k;
  k=a+t;
  return(k);    <- أمر ارجاع الدالة
}

int min(int c1, int c2)
{ int t;
  if(c1<c2)
  t=c1;
  else t=c2;
  return(t);
}
```

Subject Date

```
void show ( )
{
  int i ;
 for(i=1 ; i<=5 ;++i)
  cout<<"\n Borland C++";
}
```

- ملاحظة: يمكن للبارامترات في الدالة أن تكون مصفوفات أيضاً.
- مثال: باستخدام مفهوم الدالة بالمصفوفة أكتب برنامج يقوم بإدخال عناصر مصفوفة وطباعة عناصر هذه المصفوفة.

```
#include <iostream.h>
void show (int X[5]);
void main ( )
{ int a[5], i ;
 for(i=0 ; i<=5 ; ++i)
   cin>>a[i];
  show(a);
```

اسم المصفوفة للطباعة فقط بدون

```
}
void show(int X [5])
 { int j;
   for( j=0 ; j<5 ; ++j)
   cout<< X[j];
 }
```

تمرين: باستخدام دالة المصفوفة أكتب برنامج يحسب:
(أ) مجموع عناصر المصفوفة ذات بعد واحد. (2) جمع مصفوفتين ذات بعد واحد

تمرين: باستخدام المصفوفات أكتب برنامج يبادل بين عناصر مصفوفتين.

Subject: "المحاضرة السابعة" Date:

المؤشرات : Pointer

0 1 2 ... 100 101 102 103

int X ;

X = 37 ;

كما هو معلوم : ان الذاكرة يمكن تمثيلها على شكل شريط مقسم الى بايتات ، كل بايت في الذاكرة له رقم معين ، ترقيم البايتات يبدأ من الرقم صفر .

والتي تمثل بالشكل السابق .

نأخذ المثال التالي :

int X ;

X = 37 ;

هذا يعني الإعلان : التصريح أو حجز عنوان في الذاكرة مقداره (2) بايت ويسمى X .

أما (X = 37) تمثل محتوى العنوان وهي القيمة التي يتم تخزينها داخل عنوان المتغير X .

لابد من التمييز بين العنوان وهو رقم البايت في الذاكرة وبين محتوى العنوان .

تعريف المؤشر :

عند تعريف المؤشر نستخدم الإشارة (*) والتي تمثل بالشكل التالي :

نوع المتغيرات ـ فراغ ـ * ـ اسم المؤشر ;

* : عامل العنونة غير المباشرة .

~~تعريف المؤشر ...~~

مثال :

int * iptr ;

(iptr = اسم المؤشر)

iptr : هو عبارة عن مؤشر يؤشر الى متغير آخر في الذاكرة بحيث هذا المتغير من النوع int ، باختصار نقول (iptr) مؤشر يؤشر الى int .

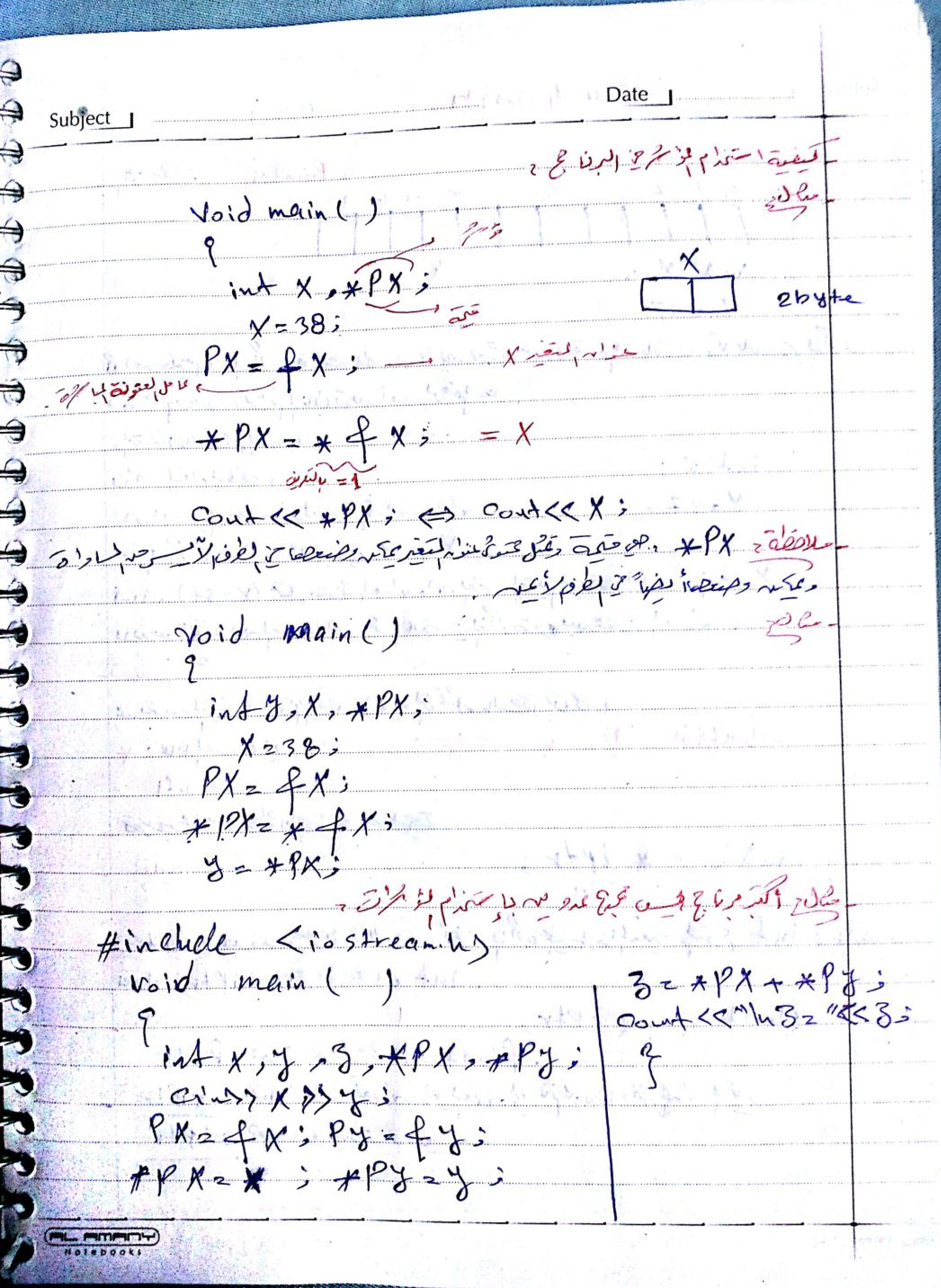
مثال :

float * fptr ;

fptr : هو مؤشر يؤشر الى float .

*iptr : تمثل محتوى العنوان وهي القيمة التي يتم تخزينها داخل عنوان المتغير .

iptr : هو عبارة عن مؤشر لمتغير .



Subject ______ Date ______

كيفية استخدام المؤشرات في البرنامج؟

مثال:

```
void main( )
{
  int x, *PX;      // PX ← مؤشر ، قيمة
  X=38;
  PX = &X;         // → عنوان المتغير X
  *PX = *&X;       // = X
  cout << *PX;  ⇔  cout << X;
```

x → [|] 2byte

& ← عامل العنونة المباشرة

*& = 1 بالتعريف

ملاحظة: *PX ، هي قيمة وتمثل محتوى عنوان المتغير ويمكن وضعها في الطرف الأيسر من المساواة ويمكن وضعها أيضاً في الطرف الأيمن.

مثال:

```
void main( )
{
  int y, X, *PX;
  X=38;
  PX=&X;
  *PX=*&X;
  y=*PX;
```

مثال: اكتب برنامج يحسب مجموع عددين باستخدام المؤشرات.

```
#include <iostream.h>
void main( )
{
  int x, y, z, *PX, *Py;
  cin>> x >> y;
  PX=&X; Py=&y;
  *PX=*; *Py=y;
  z=*PX + *Py;
  cout<<"\n z="<<z;
}
```

Subject Date

أهمية المؤشرات:

- يمكن في الواقع كتابة البرامج التطبيقية بدون استخدام المؤشرات ولكن أيضاً كتابة البرامج التطبيقية باستخدام المؤشرات.
- المؤشرات تمكن الوصول المباشر إلى عناوين المتغيرات وعناوين المصفوفات والسلاسل الحرفية بالإضافة إلى التحكم بالذاكرة الديناميكية.

من عناوين بعد تنفيذ البرنامج.

- ملاحظة: عند تعريف الدوال يمكن للبارامترات التي في الدالة أن تكون مؤشرات أيضاً، في هذه الحالة يسبق كل بارامتر الإشارة (*) أما عند استدعاء الدالة من داخل البرنامج الرئيسي يسبق البارامتر بإشارة العنوان (&)
- مثال:

```
int MM(int *x, int *y);
        (مؤشر)
t = MM(&x, &y);
      (عنوان)
```

- مثال: باستخدام مفهوم الدالة والمؤشر اكتب برنامج لجمع عددين صحيحين.

```
#include <iostream.h>
int Sum(int *a, int *b);      [illegible]
void main( )
{
  int x, y, z;
  cin>>x>>y;
  z = sum(&x, &y);      → استدعاء دالة
  cout<<"\nz="<<z;
}
int sum(int *a, int *b);
{ int c;
  c = *a + *b;
  return(c);
}
```

Subject Date

ملاحظة: أثناء إستدعاء الدالة من البرنامج الرئيسي يمكن تمرير البارامترات .

① - تمرير البارامترات بالقيمة *

② تمرير البارامترات بالعنوان &

③ - " " بالمؤشر * .

مثال: أكتب برنامج لجمع عددين صحيحين عن طريق تمرير البارامترات بالعنوان والقيمة وبالمؤشر .

- التمرير بالقيمة:

```
#include <iostream.h>
int sum(int a,int b);
    void main( )
{
  int x,y,z;
    cin>> x>>y;
    z=sum(x,y); cout<<z;
}
      int sum (int a,int b)
        int c;
        c=a+b;
        return(c);
    }
```

تمرير البارامترات بالعنوان:

```
#include <iostream.h>
void sum (int a,int b,int &c);
 {
   int x,y,z;
   cin>>x>>y;
   sum(x,y,z); cout<<z; }
void sum(int a,int b,int &c)
 { c=a+b; }
```

إرجاع مجموع القيمة

Subject ⌋ Date ⌋

مثال: أكتب برنامج يحسب مربع عدد صحيح يحسب مربع عدد صحيح عن طريق تحريك البارامترات بالقيمة

1- بالقيمة 2- بالعنوان 3- بالمؤشر

```
#include <iostream.h>
int sq( int a );
  void main( )
  { int x,y;
    cin>>x;
    y=sq(x);
    cout<<y;
  }

    int sq(int a)
     { int b;
         b=a*a;
       return(b);
     }
```

تحريك البارامتر بالمؤشر:

```
#include <iostream.h>
void sq(int*a);
  void main( )
  { int x,y;
    cin>>x;
    sq(&x);      ← استدعاء   (& = العنوان)
    cout<<x;
  }

    void sq( int *a)
      {
    *a= *a * *a;    (* = ضرب)
      }
```

void لا يرجع return يرجع

AL AMANY Notebooks

Subject | Date |

بالعنوان :

```
#include < —— >
void sq(int & a);
void main ()
{ int x;
  cin>>x;
  sq(x);        ← يستدعي قيمة X.
  cout<<x;
}
void sq(int&a)
{
  int b;
  b = a*a;
}
```

وظيفة: باستخدام الدالة والمؤشرات برنامج لقيمة الصغرى لعددين صحيحين.
(2)- مكعب عدد صحيح (3) جمع عددين صحيحين بعد زيادة كل منهما بمقدار 4.